MZUMBE UNIVERSITY
FACULTY OF SCIENCE AND TECHNOLOGY
(FST)
DEPARTMENT OF COMPUTING SCIENCE STUDIES
(CSS)

PROGRAMME:	BSc. ITS & ICT - M
COURSE CODE:	CSS 215
COURSE NAME:	DISCRETE MATHEMATICS
INSTRUCTOR:	DR. N. NYERERE
TASK:	GROUP ASSIGNMENT
SUBMISSION:	JANUARY 27th, 2022

GROUP MEMBERS: GROUP NUMBER FOUR (4)

S/N	STUDENT NAME	REGISTRATION NUMBER
1.	JOHARIA SEIF MAKULA	14322001/T.20 (ITS)
2.	IRENE PETER MAGANGA	14322003/T.20 (ITS)
3.	GOODLUCK G. MWAKYUSA	14322007/T.20 (ITS)
4.	GEORGE E. SENG'ANA	14322008/T.20 (ITS)
5.	ISAKA MABAGALA	14322009/T.20 (ITS)
6.	BERNADETHER K. SALVATORY	14322038/T.20 (ITS)
7.	SARAFINA W. MALUNDE	14320029/T.20 (ICT - M)
8.	GODLOVE S. NKYA	13304023/T.19 (ICT - M)
9.	DANIEL M. MASUBO	14320039/T.20 (ICT - M)
10.	GODIAN J. MKILANIA	14320017/T.20 (ICT - M)
11.	NDELECHE H. NDELECHE	14320012/T.20 (ICT - M)
12.	JUMA J. MANAMBA	14320018/T.20 (ICT - M)
13.	LEAH K. NDUNGURU	14320009/T.20 (ICT - M)

QUESTION
Explain Spanning trees and Co – trees with real life example.

SPANNING TREES AND CO - TREES IN ELEMENTARY GRAPH THEORY AND NETWORKS

AREAS COVERED

Definitions and terms under spanning, minimum spanning, co - trees

SPANNING TREES

- Spanning tree
- Connected graph
- Undirected graph
- Directed graph
- Simple graph
- Edges
- Vertices
- Circles or circuits.
- Properties of spanning trees.
- Circuit ranks.
- Application of spanning trees,

MINIMUM SPANNING TREES

- Definition
- Minimum spanning tree algorithm
- Explicitly enumeration.
- Kruskal's algorithm.
- Prim's algorithm.
- Applications of minimum spanning trees.

CO-TREES

- Definition
- Trees
- Examples

CONCLUSION

SPANNING TREES

Important terms:

Spanning tree: is a connected graph using all vertices in which there are no circuits. In other words, there is a path from any vertex to any other vertex, but no circuits.

Edges: Refers to a particular type of line segment that joins two vertices (corner points) or line segment on the boundary from one point to another. (_______).

Vertices: Refers to a corner where edges meets or a point where two lines meet. (●).

Circuits or cycles: Refers to a closed path. A path from a vertex back to itself.

Connected Graph: A connected graph is a graph in which we can reach any vertex from any vertex. Thus, in a connected graph, all vertices are connected to each other through at least one path.

Undirected Graph: The edges do not have a particular direction in a graph. In an undirected graph, the edges are bidirectional.

Directed Graph: In this case, the edges are unidirectional. Thus, we can go from only one end to the other ends. For every edge, there is an associated direction.

Simple graph: A simple graph is a graph that does not contain cycles and loops.

Properties of Spanning Trees

Notice, that all these spanning trees have something in common. In any spanning tree of nodes, there will always be n-1 edges. For a tree to be a spanning tree, it must connect all the nodes, but not form any loops or cycles.

Also known as circuit rank as elaborated below:

Circuit ranks

Circuit rank is the minimum number of edges that must be removed from the graph to break all its cycles, making it into a tree or forest.

Let ‘G’ be a connected graph with ‘n’ vertices and ‘m’ edges. A spanning tree ‘T’ of G contains (n-1) edges.

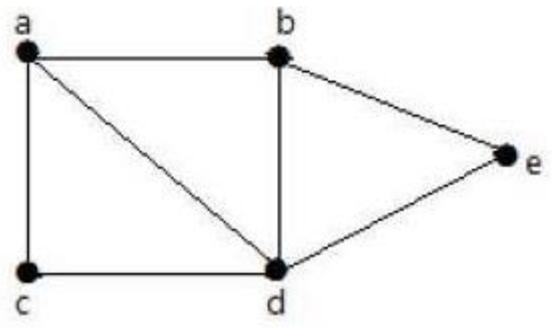
Therefore, the number of edges you need to delete from ‘G’ in order to get a spanning tree = **m - (n-1)**, which is called the **Circuit rank** of G.

This formula is true, because in a spanning tree you need to have ‘**n - 1**’ edges. Out of ‘**m**’ edges, you need to keep ‘n – 1’ edges in the graph.

Hence, deleting ‘n–1’ edges from ‘m’ gives the edges to be removed from the graph in order to get a spanning tree, which should not form a cycle.

Example

Look at the following graph



For the graph given in the above example, you have **m =7** edges and **n = 5** vertices.

Then the circuit rank is

$$\begin{aligned} \mathbf{G} &= \mathbf{m - (n - 1)} \\ &= \mathbf{7 - (5 - 1)} \\ &= \mathbf{3} \end{aligned}$$

Application of spanning trees in real life:

i) For implementing Routing protocols in computer networks.

It used for build a loop – free logical topology for Ethernet networks, also avoid bridge loops when redundant paths exist.

ii) In civil network planning to build networks.

iii) For clustering i.e. grouping, similar objects under one category and distinguishing from other categories.

MINIMUM SPANNING TREES

Definition:

A spanning tree for which the sum of the edge weights is minimum. Or

A minimum spanning tree is defined for a weighted graph. A spanning tree having minimum weight is defined as a minimum spanning tree. This weight depends on the weight of the edges.

In real-world applications, the weight could be the distance between two points, cost associated with the edges or simply an arbitrary value associated with the edges.

Minimum Spanning Tree Algorithms

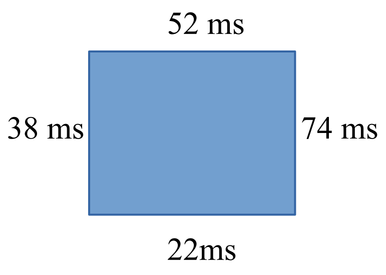
There are three minimum spanning tree algorithms namely: explicitly enumeration, Kruskal's algorithm and Prim's algorithm. These algorithms can be applied depending on the problem at hand that is to be solved.

Consider the below Hannah's problem

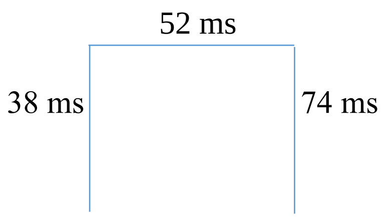
All computers must be connected to the Internet or to another computer that is connected to the Internet. This forms a spanning tree of the University's computers at Hannah's university. Any spanning tree works, but with delays on every hop, some paths are faster than others. Hannah wants to choose a spanning tree, which will assure the fastest or close to the fastest overall system performance. This is the external criteria, which determines the best spanning tree.

1. **Explicitly enumeration**

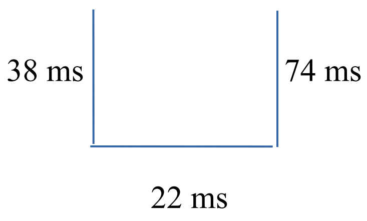
   For a small network, as shown in the Network and Spanning Trees Diagram, we can find the best spanning tree by explicit enumeration that is writing them all out and calculating their cost to obtain the best one. For example, here is a network with four nodes and four edges. We assign weights, in the form of delays, for each edge. We can make four different spanning trees for this network. These are illustrated in the Network and Spanning Tree diagram.



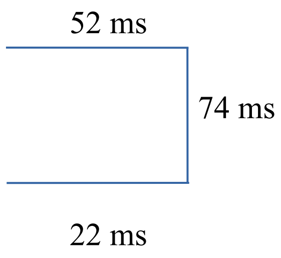
A four node, four (4) edged Network



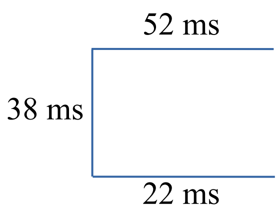
Spanning tree 1



Spanning tree 2



Spanning tree 3



Spanning tree 4

Spanning tree	**Delay**	**Delay**	**Delay**	**Delay**	**Total/Sum**
1		38	52	74	164
2	22	38		74	134
3	22		52	74	148
4	22	38	52		112

With only four spanning trees to choose from, we can calculate the total weights for each proposed spanning tree, and select the lowest total. In the Spanning Tree Table above, we see that Spanning Tree 4 has the lowest total. That is the tree we will suggest for Hannah's problem to be solved.

2. Kruskal's algorithm

Kruskal's algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using, which we can traverse, every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.

How does Kruskal's algorithm work?

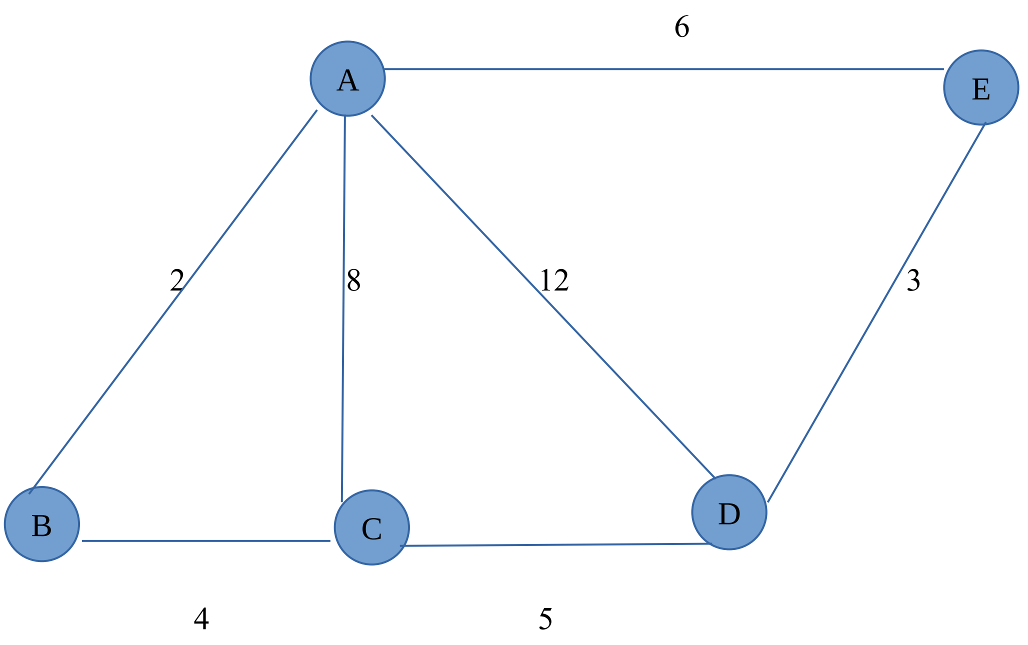
In Kruskal's algorithm, we start from edges with the lowest weight and keep adding the edges until the goal is reached. The steps to implement Kruskal's algorithm are listed as follows:

- First, sort all the edges from low weight to high.
- Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle, then reject the edge.
- Continue to add the edges until we reach all vertices, and a minimum spanning tree is created. The applications of Kruskal's algorithm are:
- Kruskal's algorithm can be used to layout electrical wiring among cities.
- It can be used to lay down LAN connections.

Example of Kruskal's algorithm

Now, let us see the working of Kruskal's algorithm using an example. It will be easier to understand Kruskal's algorithm using an example.

Suppose a weighted graph is:



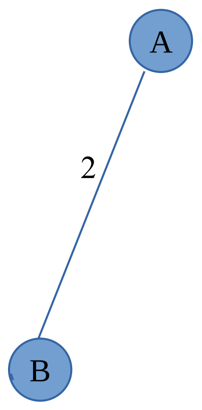
Obtain those data from the graph above as follows

Edge	**AB**	**AC**	**AD**	**AE**	**BC**	**CD**	**DE**
Weight	2	8	12	6	4	5	3

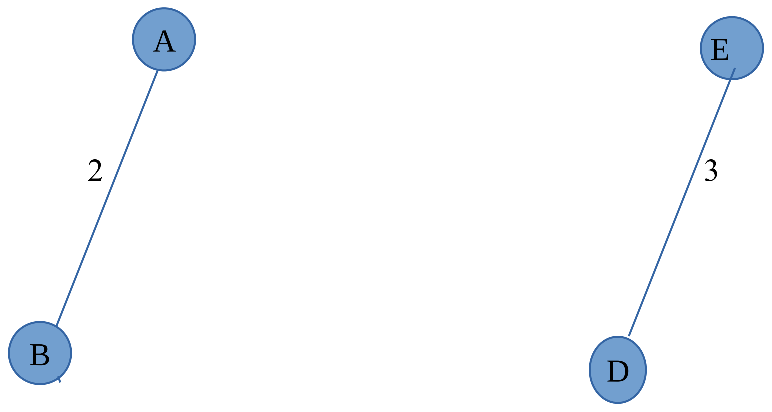
Sorting the data in ascending order becomes

Edge	**AB**	**DE**	**BC**	**CD**	**AE**	**AC**	**AD**
Weight	2	3	4	5	6	8	12

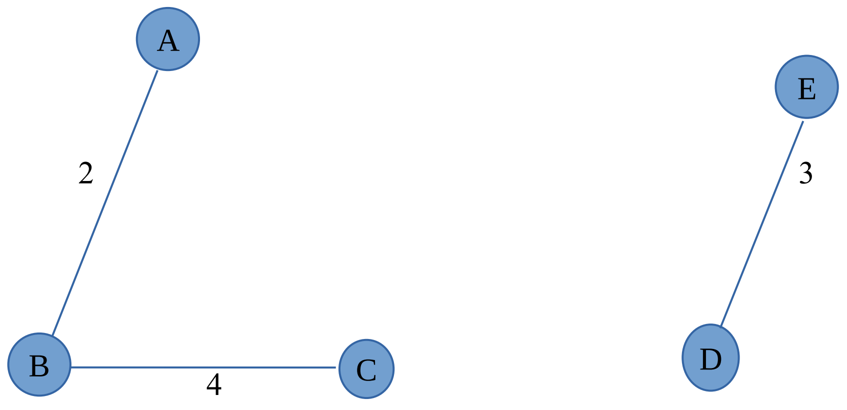
Steps undertaken to construct the Minimum Spanning Tree (MST) are as follows:
Step 1: First Add the Edge **AB** with weight 2 to the MST.



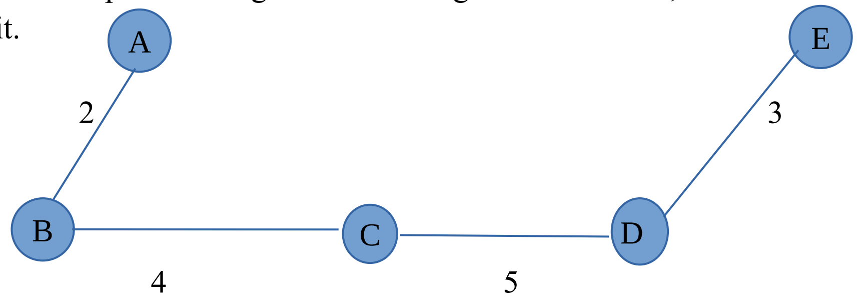
Step 2: Add the Edge **DE** with weight 3 to the MST, since it does not create a cycle or circuit.



Step 3: Add the Edge **BC** with weight 4 to the MST, since it does not create a cycle or circuit.



Step 4: Then pick the Edge CD with weight 5 to the MST, since it does not create a cycle or circuit.

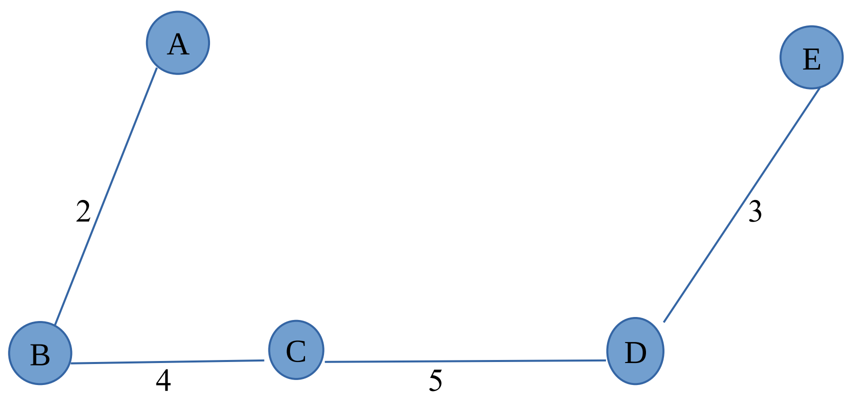


Step 5: Afterwards pick the Edge AE with weight 6 to the MST but including it will lead to a cycle so we ignore it.

Step 6: Afterwards pick the Edge AC with weight 8 to the MST but including it will lead to a cycle so we ignore it.

Step 7: Afterwards pick the Edge AD with weight 12 to the MST but including it will lead to a cycle so we ignore it.

Thus, the final MST from the above given weighted graph by using Kruskal's algorithm is



Therefore the cost of the **MST = AB+BC+CD+DE**

=2+4+5+3

=14. Answer

3. Prim's algorithm

Prim's algorithms a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

How does the prim's algorithm work?

Prim's algorithm is a greedy algorithm that starts from one vertex and continue to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows:

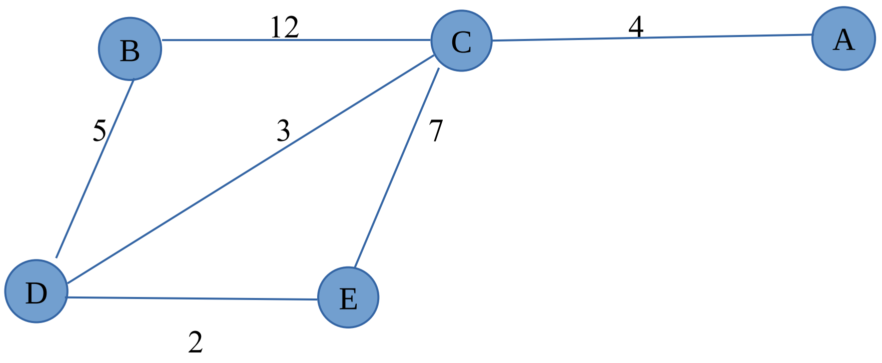
- First, we have to initialize an MST with the randomly chosen vertex.
- Now, we have to find all the edges that connect the tree in the above step with the new vertices. From the edges found, select the minimum edge and add it to the tree.
- Repeat step 2 until the minimum spanning tree is formed.

The applications of prim's algorithm in real life are:

- Prim's algorithm can be used in network designing.
- It can be used to make network cycles.
- It can also be used to lay down electrical wiring cables.

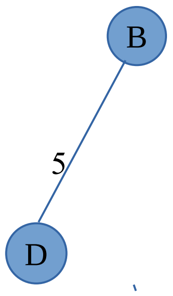
Example of Prim's algorithm

Now, let us see the working of Prim's algorithm using an example. It will be easier to understand the Prim's algorithm using an example. Suppose the weighted graph below:

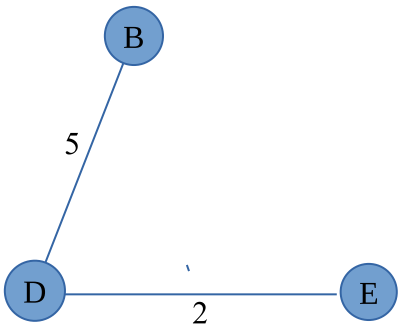


Steps undertaken to construct the Minimum Spanning Tree (MST) are as follows:
Step 1: First we have to choose a vertex from graph given above, we chose vertex B.

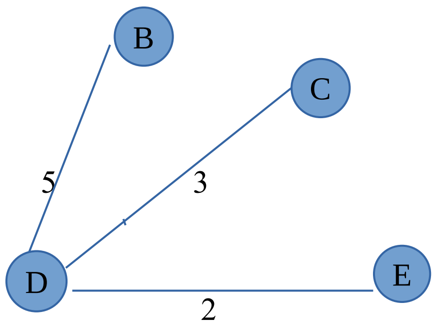
Step 2: Now we have to choose and add the shortest edges from vertex B. However, from B two edges are B to D, with weight 5 and B to C with weight 12. Among the edges, the edge BD has minimum weight, so we add it to the MST.



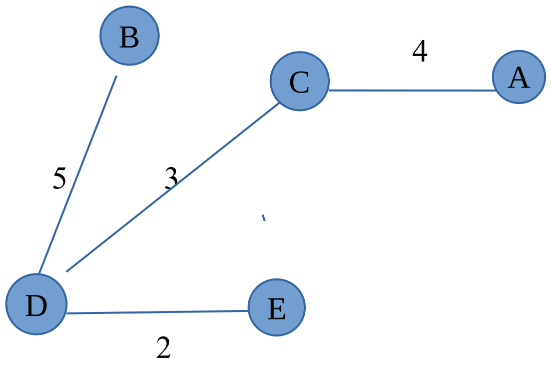
Step 3: Then, again choose edge with minimum weight among all other edges. In this case, edge **DE** and **CD** are such edges.



Step 4: Now, select edge **CD** and add it to the MST.



Step 5: Now choose edge CA, here we cannot select the edge CE as it would create a cycle or circuit to the graph. Therefore, we choose the edge CA and add it to the MST.



Therefore, the graph is produced by only following five steps in the MST of the given graph above. **The cost of MST = 5+2+3+4 = 14 units**.

Applications of minimum spanning trees

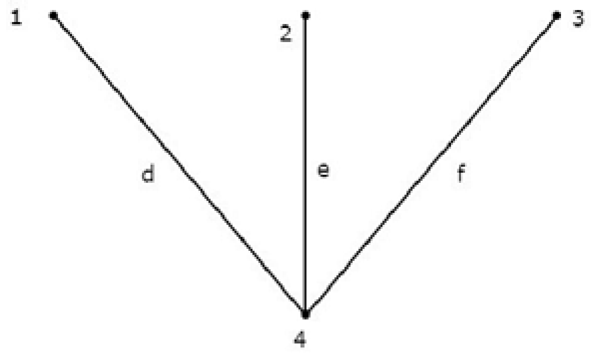
- In designing telecommunication networks and water supply networks.
- For finding paths in a map.
- In electrical grid systems.

CO-TREES

Definition
Co-tree(T*) of a spanning tree T in a connected graph G is the spacing sub-graph of G containing exactly those edges of G which are not in T.
Or
Co-tree is a complementary set of branches of the tree. Also known as links. Links are those elements included in tree links to form sub-trees.

Trees
Tree is a connected subgraph of a given graph, which contains all the nodes of a graph. However, there should not be any loop in that subgraph. The branches of a tree are called as twigs.

Consider the following **connected subgraph** of the graph, which is shown in the Example of the beginning of this chapter.



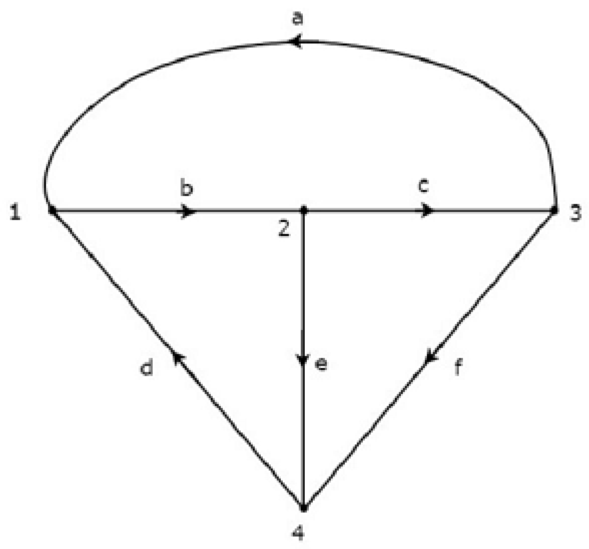
This connected subgraph contains all the four nodes of the given graph and there is no loop. Hence, it is a **Tree**.

This tree has only three branches out of six branches of given graph, Because, if we consider even single branch of the remaining branches of the graph, then there will be a loop in the above connected subgraph. Then, the resultant connected subgraph will not be a Tree.

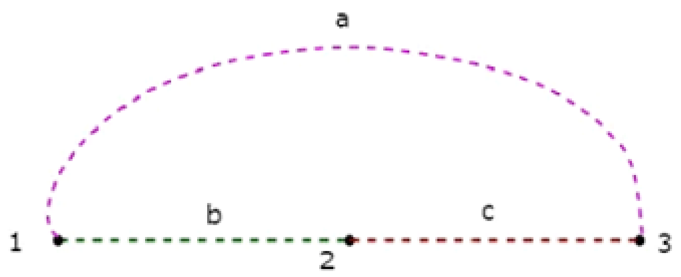
From the above Tree, we can conclude that the **number of branches** that are present in a Tree should be equal to ***n* – 1**, where 'n' is the number of nodes of the given graph.

Co-Tree is a sub-graph, which is formed with the branches that are removed while forming a Tree. Hence, it is called as Complement of a Tree. For every Tree, there will be a corresponding Co-Tree and its branches are called as links or chords. In general, the links are represented with dotted lines.

Examples:
Consider the graph given below:



The Co-tree corresponding to this graph is as below:



This Co-Tree has only three nodes instead of four nodes of the given graph, because Node 4 is isolated from the above Co-Tree. Therefore, the Co-Tree need not be a connected subgraph. This Co-Tree has three branches and they form a loop.

The **number of branches** that are present in a co-tree will be equal to the difference between the number of branches of a given graph and the number of twigs. Mathematically, it can be written as

$\boldsymbol{l = b - (n - 1)}$

$l = b - n + 1$

Where,

- l- Is the number of links.
- b- Is the number of branches present in the given graph.
- n- Is the number of nodes present in a given graph.

Conclusively, if you combine a Tree and its corresponding Co-Tree, then you will get the **original graph** as illustrated above.

To sum up, the main aim of spanning trees is to design networks (Example Telephone network) and finding solutions for complex problems in a simple manner.

REFERENCES

- Kenneth H. Rosen. (2007). *Discrete mathematics and its application*. Sixth edition. McGraw-Hill. New York.
- Wu, B. & Chao, K. (2004). *Spanning Trees and Optimization Problems*. Chapman and Hall/CRC.
- https://www.javatpoint.com/spanning-tree
- https://www.tutorialspoint.com/network_theory/network_theory_topology.htm